Задание №1

Вызов <https://mysite.com/one>

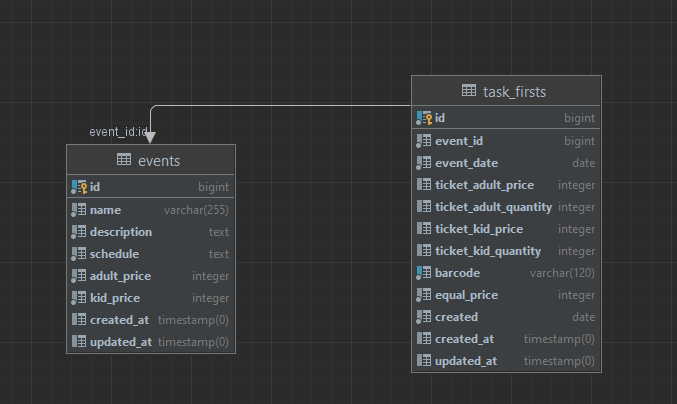
Контроллер - TaskFirstController.php

На вход получает массив данных, в цикле начинаем обработку:

Генерация barcode

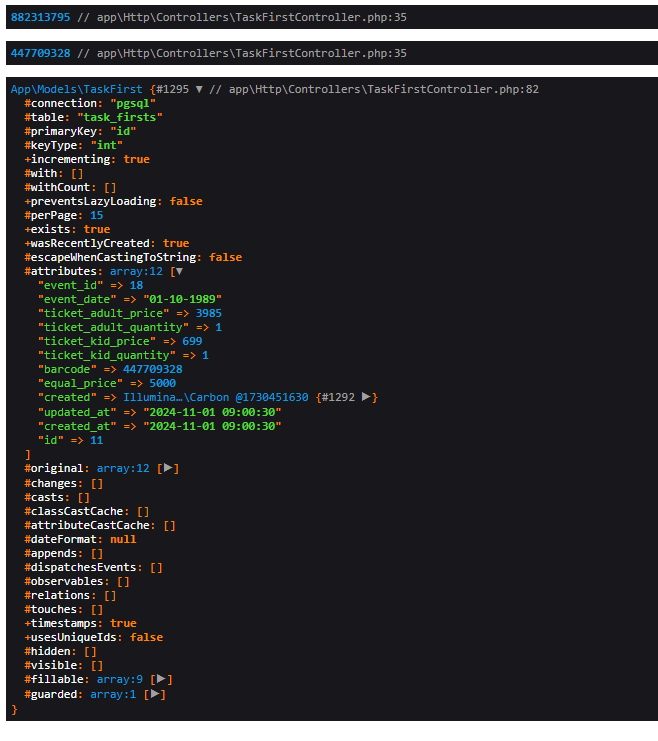
Проверка его в нашей БД, при отсутствии продолжаем, открываем транзакцию по данным и проверяем заказ, генерируем до положительного ответа API. Отправляем barcode на второе API при положительном ответе сохраняем в БД, иначе отменяем транзакцию.

Дополнительно была создана таблица “events”, для id события со связью один ко многим



Также для невозможности создание записи с дублирующим barcode, донному столбцу было назначено свойство уникального в таблице на уровне СУБД. В ТЗ ничего не сказано, на каком уровне должна быть уникальность, всей таблицы или в разрезе события. Если в разрезе события, то можно использовать составной уникальный ключ event\_id + barcode.

Работу можем посмотреть в дампе:



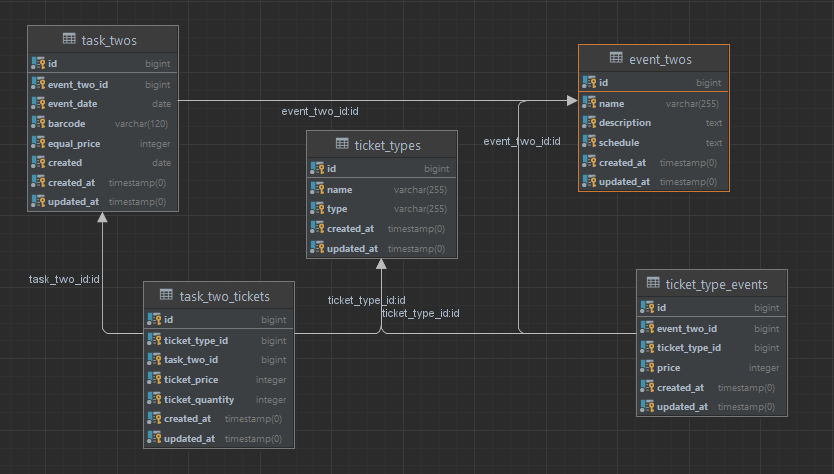
На первый barcode получен отрицательные ответ, было сгенерирован второй и его мы уже можем наблюдать в таблице

Задание 2.1

Вызов <https://mysite.com/two>

Контроллер - TaskTwoController.php

Диаграмма таблиц:



Т.к. по ТЗ сказано, что есть вероятность других типов билетов, просто дополнительные столбцы в таблицу посчитал не правильно, хотя является самым быстрым способом.

1. Была добавлена таблица «ticket\_types»

- name (Человеко читаемое название билета)

- type (префикс билета по API = ticket\_adult)

Через админку можем добавить любое количество типов билетов. В ТЗ ничего не сказано про то будет ли меняться информация по билетам, дополнительные столбцы и т.д. Если изменения более существенные по каждому типу билетов, то уже имеет смысл рассмотреть паттерн фабричный метод, выносить общую часть, наследоваться и на интерфейсе выводить какие-то методы реализации.

2. Таблица ticket\_type\_events

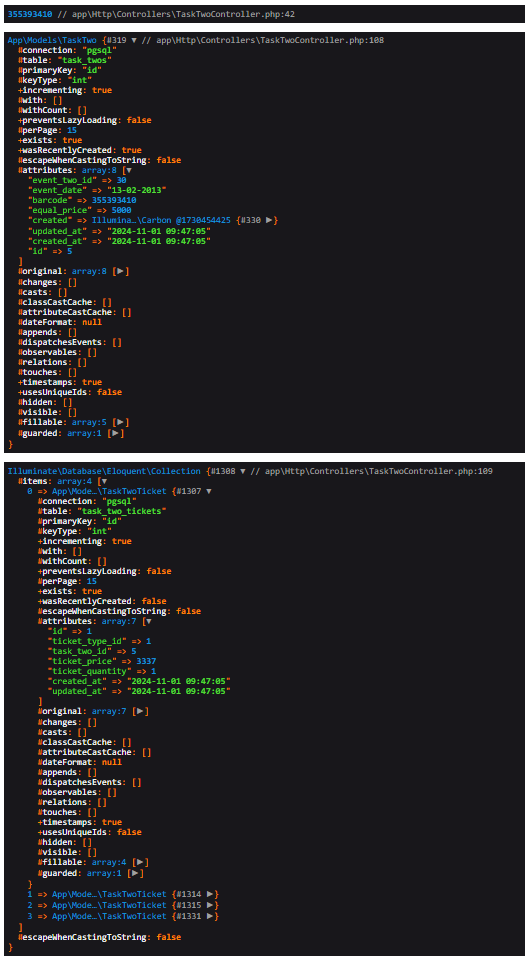
Является связующей таблицей между типом билетов и мероприятий, на каждое мероприятие создаем свой список типа биллетов. Со связью многие ко многим. Столбец price из таблицы «event\_two» был вынесен сюда

3. Таблица task\_two\_tickets

Хранит данные: тип билета, заказ, цена и количество билетов данного типа

Соответственно по связи таблиц один ко многим – заказа и билетов мы можем получить все билеты (всех типов) заказа, данные о событии также хранятся в заказе

Дамп работы:



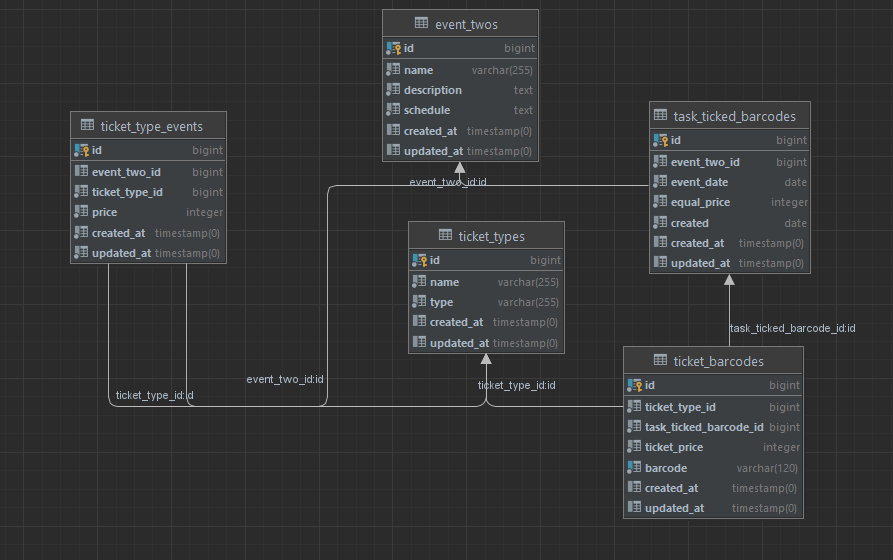
Сохранен заказ и четыре билета

Задание 2.2

Вызов <https://mysite.com/ticket>

Контроллер - TaskTicketBarcodeController.php

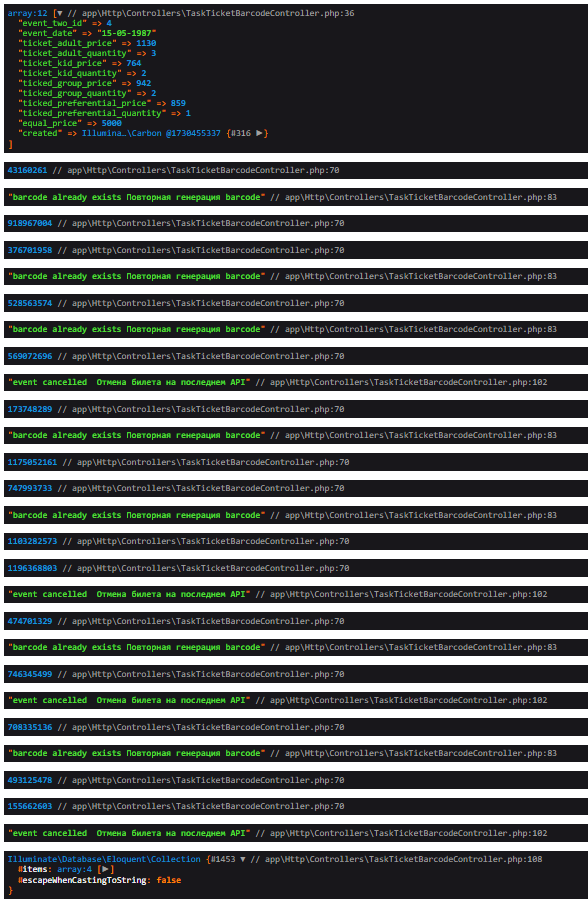
Диаграмма таблиц:



1. В таблице заказов task\_ticked\_barcodes, оставлены только поля 'event\_two\_id','event\_date','equal\_price','created', все остальное вынесено в таблицу ticket\_barcodes
2. В таблице ticket\_barcodes, убран столбец ticket\_quantity, т.к. количество теперь не имеет смысл указывать и оно не может превышаться единицы и добавлен столбец barcode, для каждого билета. Связь таблиц с заказом осталась прежней. Логику работу можно посмотреть в контроллере

Дамп работы:

По нему видно что из входных данных восемь билетов на последнем API прошли проверку только четыре.



В ТЗ также не указанно сохранять ли заказ и билеты не прошедшие проверку с каким-то статусом. Ничего не сказано про работу API с другой стороны. Не сделана валидация поступающих данных, т.к. написано API не реализовывать.

<https://github.com/s-kholman/bitbucket>

Установка:

git clone <https://github.com/s-kholman/bitbucket>

composer install

Добавить данные в

.ENV

DB\_DATABASE=

DB\_USERNAME=

php artisan migrate

php artisan key:generate

php artisan db:seed

php artisan serve